Crate commonware_consensus

Settings

Help

Summary

Source

Order opaque messages in a Byzantine environment.

Status

`commonware-consensus` is ALPHA software and is not yet recommended for production use. Developers should expect breaking changes and occasional instability.

Modules

- simplex
- Simple and fast BFT agreement inspired by Simplex Consensus.

Traits

- Automaton
- Automaton is the interface responsible for driving the consensus forward by proposing new payloads and verifying payloads proposed by other participants.
- Committer
- Committer is the interface responsible for handling notifications of payload status.
- Relay
- Relay is the interface responsible for broadcasting payloads to the network.
- Supervisor
- Supervisor is the interface responsible for managing which participants are active at a given time.

Type Aliases

- Activity
- Activity is specified by the underlying consensus implementation and can be interpreted if desired.

- Proof
- Proof is a blob that attests to some data.

Module simplex

Settings

Help

Summary

Source

Simple and fast BFT agreement inspired by Simplex Consensus.

Inspired by Simplex Consensus, `simplex` provides simple and fast BFT agreement that seeks to minimize view latency (i.e. block time) and to provide optimal finalization latency in a partially synchronous setting.

Features

- Wicked Fast Block Times (2 Network Hops)
- Optimal Finalization Latency (3 Network Hops)
- Externalized Uptime and Fault Proofs
- Decoupled Block Broadcast and Sync
- Flexible Block Format

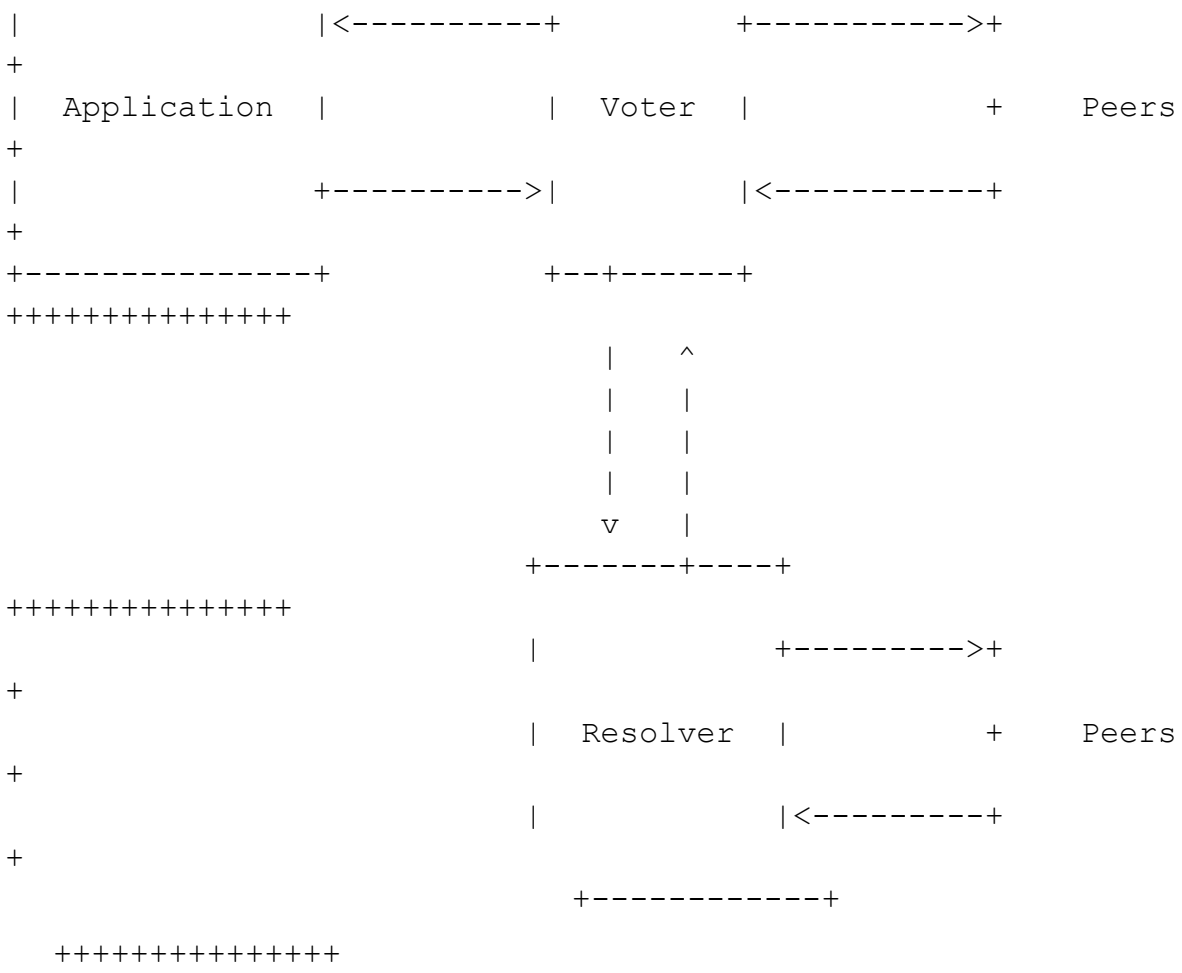
Design

Architecture

All logic is split into two components: the `Voter` and the `Resolver` (and the user of `simplex` provides `Application`). The `Voter` is responsible for participating in the latest view and the `Resolver` is responsible for fetching artifacts from previous views required to verify proposed blocks in the latest view.

To provide great performance, all interactions between `Voter`, `Resolver`, and `Application` are non-blocking. This means that, for example, the `Voter` can continue processing messages while the `Application` verifies a proposed block or the `Resolver` verifies a notarization.

```
+---------------+                +----------+
+++++++++++++++
```



Application is usually a single object that implements the `Automaton`, `Relay`, `Committer`, and `Supervisor` traits.

Joining Consensus

As soon as `2f+1` votes or finalizes are observed for some view `v`, the `Voter` will enter `v+1`. This means that a new participant joining consensus will immediately jump ahead to the latest view and begin participating in consensus (assuming it can verify blocks).

Persistence

The `Voter` caches all data required to participate in consensus to avoid any disk reads on on the critical path. To enable recovery, the `Voter` writes valid messages it receives from consensus and messages it generates to a write-ahead log (WAL) implemented by `Journal`. Before sending a message, the `Journal` sync is invoked to prevent inadvertent Byzantine behavior on restart (especially in the case of unclean shutdown).

Protocol Description

Specification for View `v`

Upon entering view `v`:

- Determine leader `l` for view `v`
- Set timer for leader proposal `t_l = 2Δ` and advance `t_a = 3Δ`
    - If leader `l` has not been active (no votes) in last `r` views, set `t_l` to 0.
- If leader `l`, broadcast `notarize(c,v)`
    - If can't propose container in view `v` because missing notarization/nullification for a previous view `v_m`, request `v_m`

Upon receiving first `notarize(c,v)` from `l`:

- Cancel `t_l`
- If the container's parent `c_parent` is notarized at `v_parent` and we have null notarizations for all views between `v` and `v_parent`, verify `c` and broadcast `notarize(c,v)`

Upon receiving `2f+1 notarize(c,v)`:

- Cancel `t_a`
- Mark `c` as notarized
- Broadcast `notarization(c,v)` (even if we have not verified `c`)
- If have not broadcast `nullify(v)`, broadcast `finalize(c,v)`
- Enter `v+1`

Upon receiving `2f+1 nullify(v)`:

- Broadcast `nullification(v)`
    - If observe `>= f+1 notarize(c,v)` for some `c`, request `notarization(c_parent, v_parent)` and any missing `nullification(*)` between `v_parent` and `v`. If `c_parent` is than last finalized, broadcast last finalization instead.
- Enter `v+1`

Upon receiving `2f+1 finalize(c,v)`:

- Mark `c` as finalized (and recursively finalize its parents)
- Broadcast `finalization(c,v)` (even if we have not verified `c`)

Upon `t_l` or `t_a` firing:

- Broadcast `nullify(v)`
- Every `t_r` after `nullify(v)` broadcast that we are still in view `v`:

    - Rebroadcast `nullify(v)` and either `notarization(v-1)` or `nullification(v-1)`

Deviations from Simplex Consensus

- Fetch missing notarizations/nullifications as needed rather than assuming each proposal contains a set of all notarizations/nullifications for all historical blocks.
- Introduce distinct messages for `notarize` and `nullify` rather than referring to both as a `vote` for either a “block” or a “dummy block”, respectively.
- Introduce a “leader timeout” to trigger early view transitions for unresponsive leaders.
- Skip “leader timeout” and “notarization timeout” if a designated leader hasn’t participated in some number of views (again to trigger early view transition for an unresponsive leader).
- Introduce message rebroadcast to continue making progress if messages from a given view are dropped (only way to ensure messages are reliably delivered is with a heavyweight reliable broadcast protocol).

Structs

- Config
- Configuration for the consensus engine.
- Context
- Context is a collection of metadata from consensus about a given payload.
- Engine
- Instance of `simplex` consensus engine.
- Prover
- Encode and decode proofs of activity.

Enums

- Error
- Errors that can occur during consensus.

Constants

- CONFLICTING_FINALIZE
- Finalize a payload that conflicts with a previous finalize.
- CONFLICTING_NOTARIZE

- Notarize a payload that conflicts with a previous notarize.
- FINALIZE
- Finalize a payload at a given view.
- NOTARIZE
- Notarize a payload at a given view.
- NULLIFY_AND_FINALIZE
- Nullify and finalize in the same view.

Type Aliases

- View
- View is a monotonically increasing counter that represents the current focus of consensus.

Trait Automaton Copy item path

Settings

Help

Summary

Source

```
pub trait Automaton:
    Clone
    + Send
    + 'static {
    type Context;

    // Required methods
    fn genesis(&mut self) -> impl Future<Output = Digest> +
Send;

   fn propose(
        &mut self,
        context: Self::Context,
    ) -> impl Future<Output = Receiver<Digest>> + Send;

   fn verify(
        &mut self,
        context: Self::Context,
        payload: Digest,
```

```
    ) -> impl Future<Output = Receiver<bool>> + Send;
}
```

Automaton is the interface responsible for driving the consensus forward by proposing new payloads and verifying payloads proposed by other participants.

Required Associated Types

Source

```
type Context
```

Context is metadata provided by the consensus engine to associated with a given payload.

This often includes things like the proposer, view number, the height, or the epoch.

Required Methods

Source

```
fn genesis(&mut self) -> impl Future<Output = Digest> + Send
```

Payload used to initialize the consensus engine.

Source

```
fn propose(
    &mut self,
    context: Self::Context,
) -> impl Future<Output = Receiver<Digest>> + Send
```

Generate a new payload for the given context.

If it is possible to generate a payload, the Digest should be returned over the provided channel. If it is not possible to generate a payload, the channel can be dropped. If construction takes too long, the consensus engine may drop the provided proposal.

Source

```
fn verify(
    &mut self,
    context: Self::Context,
    payload: Digest,
) -> impl Future<Output = Receiver<bool>> + Send
```

Verify the payload is valid.

If it is possible to verify the payload, a boolean should be returned indicating whether the payload is valid. If it is not possible to verify the payload, the channel can be dropped.

Dyn Compatibility

This trait is not dyn compatible.

In older versions of Rust, dyn compatibility was called "object safety", so this trait is not object safe.

Implementors

Trait Committer

Settings

Help

Summary

Source

```
pub trait Committer:
    Clone
    + Send
    + 'static {
    // Required methods
    fn prepared(
        &mut self,
        proof: Proof,
        payload: Digest,
    ) -> impl Future<Output = ()> + Send;
```

```
    fn finalized(
         &mut self,
         proof: Proof,
         payload: Digest,
     ) -> impl Future<Output = ()> + Send;
}
```

Committer is the interface responsible for handling notifications of payload status.

Required Methods

Source

```
fn prepared(
    &mut self,
    proof: Proof,
    payload: Digest,
) -> impl Future<Output = ()> + Send
```

Event that a payload has made some progress towards finalization but is not yet finalized.

This is often used to provide an early (“best guess”) confirmation to users.

Source

```
fn finalized(
    &mut self,
    proof: Proof,
    payload: Digest,
) -> impl Future<Output = ()> + Send
```

Event indicating the container has been finalized.

Dyn Compatibility

This trait is not dyn compatible.

In older versions of Rust, dyn compatibility was called "object safety", so this trait is not object safe.

Implementors

Trait Relay

Settings

Help

Summary

Source

```
pub trait Relay:
    Clone
    + Send
    + 'static {
    // Required method
    fn broadcast(&mut self, payload: Digest) -> impl
Future<Output = ()> + Send;
}
```

Relay is the interface responsible for broadcasting payloads to the network.

The consensus engine is only aware of a payload's digest, not its contents. It is up to the relay to efficiently broadcast the full payload to other participants.

Required Methods

Source

```
fn broadcast(&mut self, payload: Digest) -> impl Future<Output
= ()> + Send
```

Called once consensus begins working towards a proposal provided by `Automaton` (i.e. it isn't dropped).

Other participants may not begin voting on a proposal until they have the full contents, so timely delivery often yields better performance.

Dyn Compatibility

This trait is not dyn compatible.

In older versions of Rust, dyn compatibility was called "object safety", so this trait is not object safe.

Implementors

Trait Supervisor

Copy item path

Settings

Help

Summary

Source

```
pub trait Supervisor:
    Clone
    + Send
    + 'static {
    type Index;
    type Seed;

    // Required methods
    fn leader(&self, index: Self::Index, seed: Self::Seed) ->
Option<PublicKey>;

   fn participants(&self, index: Self::Index) ->
Option<&Vec<PublicKey>>;

   fn is_participant(
        &self,
        index: Self::Index,
        candidate: &PublicKey,
    ) -> Option<u32>;

   fn report(
        &self,
        activity: Activity,
        proof: Proof,
    ) -> impl Future<Output = ()> + Send;
}
```

Supervisor is the interface responsible for managing which participants are active at a given time.

Synchronization

It is up to the user to ensure changes in this list are synchronized across nodes in the network at a given `Index`. If care is not taken to do this, consensus could halt (as different participants may have a different view of who is active at a given time).

The simplest way to avoid this complexity is to use a consensus implementation that reaches finalization on application data before transitioning to a new `Index` (i.e. Tendermint).

Implementations that do not work this way (like `simplex`) must introduce some synchrony bound for changes (where it is assumed all participants have finalized some previous set change by some point) or "sync points" (i.e. epochs) where participants agree that some finalization occurred at some point in the past.

Required Associated Types

Source

```
type Index
```

Index is the type used to indicate the in-progress consensus decision.

Source

```
type Seed
```

Seed is a consensus artifact to use as randomness for leader selection.

Required Methods

Source

```
fn leader(&self, index: Self::Index, seed: Self::Seed) ->
Option<PublicKey>
```

Return the leader at a given index for the provided seed.

Source

```
fn participants(&self, index: Self::Index) ->
Option<&Vec<PublicKey>>
```

Get the sorted participants for the given view. This is called when entering a new view before listening for proposals or votes. If nothing is returned, the view will not be entered.

Source

```
fn is_participant(
    &self,
    index: Self::Index,
    candidate: &PublicKey,
) -> Option<u32>
```

Source

```
fn report(
    &self,
    activity: Activity,
    proof: Proof,
) -> impl Future<Output = ()> + Send
```

Report some activity observed by the consensus implementation.

Dyn Compatibility

This trait is not dyn compatible.

In older versions of Rust, dyn compatibility was called "object safety", so this trait is not object safe.

Implementors

commonware_consensus::simplex

Struct Config

Settings

Help

Summary

Source

```
pub struct Config<C: Scheme, H: Hasher, A: Automaton<Context = Context>, R: Relay, F: Committer, S: Supervisor<Seed = (), Index = View>> {Show 19 fields}
```

Configuration for the consensus engine.

Fields

`crypto: C`

Cryptographic primitives.

`hasher: H`

Hashing algorithm.

`automaton: A`

Automaton for the consensus engine.

`relay: R`

Relay for the consensus engine.

`committer: F`

Committer for the consensus engine.

`supervisor: S`

Supervisor for the consensus engine.

`registry: Arc<Mutex<Registry>>`

Prometheus metrics registry.

`mailbox_size: usize`

Maximum number of messages to buffer on channels inside the consensus engine before blocking.

`namespace: Vec<u8>`

Prefix for all signed messages to prevent replay attacks.

`replay_concurrency: usize`

Number of views to replay concurrently during startup.

`leader_timeout: Duration`

Amount of time to wait for a leader to propose a payload in a view.

`notarization_timeout: Duration`

Amount of time to wait for a quorum of notarizations in a view before attempting to skip the view.

`nullify_retry: Duration`

Amount of time to wait before retrying a nullify broadcast if stuck in a view.

`activity_timeout: View`

Number of views behind finalized tip to track activity derived from validator messages.

`fetch_timeout: Duration`

Timeout to wait for a peer to respond to a request.

`max_fetch_count: usize`

Maximum number of notarizations/nullifications to request/respond with at once.

`max_fetch_size: usize`

Maximum number of bytes to respond with at once.

`fetch_rate_per_peer: Quota`

Maximum rate of requests to send to a given peer.

Inbound rate limiting is handled by `commonware-p2p`.

`fetch_concurrent: usize`

Number of concurrent requests to make at once.

Implementations

Source

```
impl<C: Scheme, H: Hasher, A: Automaton<Context = Context>, R: Relay, F: Committer, S: Supervisor<Seed = (), Index = View>> Config<C, H, A, R, F, S>
```

Source

```
pub fn assert(&self)
```

Assert enforces that all configuration values are valid.

Auto Trait Implementations

```
impl<C, H, A, R, F, S> Freeze for Config<C, H, A, R, F,
S>

where
    C: Freeze,
    H: Freeze,
    A: Freeze,
    R: Freeze,
    F: Freeze,
    S: Freeze,

impl<C, H, A, R, F, S> RefUnwindSafe for Config<C, H, A,
R, F, S>

where
    C: RefUnwindSafe,
    H: RefUnwindSafe,
    A: RefUnwindSafe,
    R: RefUnwindSafe,
    F: RefUnwindSafe,
    S: RefUnwindSafe,

impl<C, H, A, R, F, S> Send for Config<C, H, A, R, F, S>

impl<C, H, A, R, F, S> Sync for Config<C, H, A, R, F, S>

where
    A: Sync,
    R: Sync,
    F: Sync,
    S: Sync,

impl<C, H, A, R, F, S> Unpin for Config<C, H, A, R, F, S>

where
    C: Unpin,
    H: Unpin,
    A: Unpin,
    R: Unpin,
    F: Unpin,
    S: Unpin,

impl<C, H, A, R, F, S> UnwindSafe for Config<C, H, A, R,
F, S>

where
```

```
    C: UnwindSafe,
    H: UnwindSafe,
    A: UnwindSafe,
    R: UnwindSafe,
    F: UnwindSafe,
    S: UnwindSafe,
```

Blanket Implementations

Source

```
impl<T> Any for T

where
    T: 'static + ?Sized,
```

Source

```
impl<T> Borrow<T> for T

where
    T: ?Sized,
```

Source

```
impl<T> BorrowMut<T> for T

where
    T: ?Sized,
```

Source

```
impl<T> Conv for T
```

Source

```
impl<T> FmtForward for T
```

Source

```
impl<T> From<T> for T
```

Source

```
impl<T> Instrument for T
```

Source

```
impl<T, U> Into<U> for T

where
    U: From<T>,
```

Source

```
impl<T> IntoEither for T
```

Source

```
impl<T> Pipe for T

where
    T: ?Sized,
```

Source

```
impl<T> Pointable for T
```

Source

```
impl<T> Same for T
```

Source

```
impl<T> Tap for T
```

Source

```
impl<T> TryConv for T
```

Source

```
impl<T, U> TryFrom<U> for T

where
```

```
    U: Into<T>,
```

Source

```
impl<T, U> TryInto<U> for T
```

```
where
    U: TryFrom<T>,
```

Source

§

```
impl<V, T> VZip<V> for T
```

```
where
    V: MultiLane<T>,
```

Source

```
impl<T> WithSubscriber for T
```

commonware_consensus::simplex

Struct Context

Settings

Help

Summary

Source

```
pub struct Context {
    pub view: View,
    pub parent: (View, Digest),
}
```

Context is a collection of metadata from consensus about a given payload.

Fields

```
view: View
```

Current view of consensus.

`parent: (View, Digest)`

Parent the payload is built on.

Payloads from views between the current view and the parent view can never be directly finalized (must exist some nullification).

Trait Implementations

Source

`impl Clone for Context`

Source

`fn clone(&self) -> Context`

Returns a copy of the value. Read more

1.0.0 · Source

`fn clone_from(&mut self, source: &Self)`

Performs copy-assignment from `source`. Read more

Auto Trait Implementations

`impl !Freeze for Context`

`impl RefUnwindSafe for Context`

`impl Send for Context`

`impl Sync for Context`

`impl Unpin for Context`

`impl UnwindSafe for Context`

Blanket Implementations

Source

`impl<T> Any for T`

```
where
    T: 'static + ?Sized,
```

Source

```
impl<T> Borrow<T> for T

where
    T: ?Sized,
```

Source

```
impl<T> BorrowMut<T> for T

where
    T: ?Sized,
```

Source

```
impl<T> CloneToUninit for T

where
    T: Clone,
```

Source

```
impl<T> Conv for T
```

Source

```
impl<T> FmtForward for T
```

Source

```
impl<T> From<T> for T
```

Source

```
impl<T> Instrument for T
```

Source

```
impl<T, U> Into<U> for T
```

```
where
    U: From<T>,
```

Source

```
impl<T> IntoEither for T
```

Source

```
impl<T> Pipe for T

where
    T: ?Sized,
```

Source

```
impl<T> Pointable for T
```

Source

```
impl<T> Same for T
```

Source

```
impl<T> Tap for T
```

Source

```
impl<T> ToOwned for T

where
    T: Clone,
```

Source

```
impl<T> TryConv for T
```

Source

§

```
impl<T, U> TryFrom<U> for T
```

```
where
    U: Into<T>,
```

Source

```
impl<T, U> TryInto<U> for T
```

```
where
    U: TryFrom<T>,
```

Source

```
impl<V, T> VZip<V> for T
```

```
where
    V: MultiLane<T>,
```

Source

```
impl<T> WithSubscriber for T
```

commonware_consensus::simplex

Struct Engine

Settings

Help

Summary

Source

```
pub struct Engine<B: Blob, E: Clock + GClock + Rng + CryptoRng + Spawner + Storage<B>, C: Scheme, H: Hasher, A: Automaton<Context = Context>, R: Relay, F: Committer, S: Supervisor<Seed = (), Index = View>> { /* private fields */ }
```

Instance of `simplex` consensus engine.

Implementations

Source

```
impl<B: Blob, E: Clock + GClock + Rng + CryptoRng + Spawner + Storage<B>, C: Scheme, H: Hasher, A: Automaton<Context = Context>, R: Relay, F: Committer, S: Supervisor<Seed = (), Index = View>> Engine<B, E, C, H, A, R, F, S>
```

Source

```
pub fn new(
    runtime: E,
    journal: Journal<B, E>,
    cfg: Config<C, H, A, R, F, S>,
) -> Self
```

Create a new `simplex` consensus engine.

Source

```
pub async fn run(
    self,
    voter_network: (impl Sender, impl Receiver),
    resolver_network: (impl Sender, impl Receiver),
)
```

Start the `simplex` consensus engine.

This will also rebuild the state of the engine from provided `Journal`.

Auto Trait Implementations

```
impl<B, E, C, H, A, R, F, S> !Freeze for Engine<B, E, C, H, A, R, F, S>
```

```
impl<B, E, C, H, A, R, F, S> !RefUnwindSafe for Engine<B, E, C, H, A, R, F, S>
```

```
impl<B, E, C, H, A, R, F, S> Send for Engine<B, E, C, H, A, R, F, S>
```

```
impl<B, E, C, H, A, R, F, S> Sync for Engine<B, E, C, H, A, R, F, S>
```

```
where
    A: Sync,
```

```
    R: Sync,
    F: Sync,
    S: Sync,
```

```
impl<B, E, C, H, A, R, F, S> Unpin for Engine<B, E, C, H, A, R, F, S>
```

```
where
    E: Unpin,
    C: Unpin,
    H: Unpin,
    A: Unpin,
    R: Unpin,
    F: Unpin,
    S: Unpin,
    <E as Clock>::Instant: Unpin,
```

```
impl<B, E, C, H, A, R, F, S> !UnwindSafe for Engine<B, E, C, H, A, R, F, S>
```

Blanket Implementations

Source

```
impl<T> Any for T

where
    T: 'static + ?Sized,
```

Source

```
impl<T> Borrow<T> for T

where
    T: ?Sized,
```

Source

```
impl<T> BorrowMut<T> for T

where
    T: ?Sized,
```

Source

```
impl<T> Conv for T
```

Source

```
impl<T> FmtForward for T
```

Source

```
impl<T> From<T> for T
```

Source

```
impl<T> Instrument for T
```

Source

```
impl<T, U> Into<U> for T

where
    U: From<T>,
```

Source

```
impl<T> IntoEither for T
```

Source

```
impl<T> Pipe for T

where
    T: ?Sized,
```

Source

```
impl<T> Pointable for T
```

Source

```
impl<T> Same for T
```

Source

§

```
impl<T> Tap for T
```

Source

```
impl<T> TryConv for T
```

Source

```
impl<T, U> TryFrom<U> for T

where
    U: Into<T>,
```

Source

```
impl<T, U> TryInto<U> for T

where
    U: TryFrom<T>,
```

Source

```
impl<V, T> VZip<V> for T

where
    V: MultiLane<T>,
```

Source

```
impl<T> WithSubscriber for T
```

commonware_consensus::simplex

Struct Prover

Copy item path

Settings

Help

Summary

Source

```
pub struct Prover<C: Scheme, H: Hasher> { /* private fields */ }
```

Encode and decode proofs of activity.

We don't use protobuf for proof encoding because we expect external parties to decode proofs in constrained environments where protobuf may not be implemented.

Implementations

Source

`impl<C: Scheme, H: Hasher> Prover<C, H>`

Source

```
pub fn new(namespace: &[u8]) -> Self
```

Create a new prover with the given signing `namespace`.

Source

```
pub fn deserialize_notarize(
    &self,
    proof: Proof,
    check_sig: bool,
) -> Option<(View, View, Digest, PublicKey)>
```

Deserialize a notarize proof.

Source

```
pub fn deserialize_notarization(
    &self,
    proof: Proof,
    max: u32,
    check_sigs: bool,
) -> Option<(View, View, Digest, Vec<PublicKey>)>
```

Deserialize a notarization proof.

Source

```
pub fn deserialize_finalize(
    &self,
    proof: Proof,
    check_sig: bool,
) -> Option<(View, View, Digest, PublicKey)>
```

Deserialize a finalize proof.

Source

```
pub fn deserialize_finalization(
    &self,
    proof: Proof,
    max: u32,
    check_sigs: bool,
) -> Option<(View, View, Digest, Vec<PublicKey>)>
```

Deserialize a finalization proof.

Source

```
pub fn deserialize_conflicting_notarize(
    &self,
    proof: Proof,
    check_sig: bool,
) -> Option<(PublicKey, View)>
```

Deserialize a conflicting notarization proof.

Source

```
pub fn deserialize_conflicting_finalize(
    &self,
    proof: Proof,
    check_sig: bool,
) -> Option<(PublicKey, View)>
```

Deserialize a conflicting finalization proof.

Source

```
pub fn deserialize_nullify_finalize(
    &self,
    proof: Proof,
    check_sig: bool,
) -> Option<(PublicKey, View)>
```

Deserialize a conflicting nullify and finalize proof.

Trait Implementations

Source

```
impl<C: Clone + Scheme, H: Clone + Hasher> Clone for Prover<C, H>
```

Source

```
fn clone(&self) -> Prover<C, H>
```

Returns a copy of the value. Read more

1.0.0 · Source

```
fn clone_from(&mut self, source: &Self)
```

Performs copy-assignment from `source`. Read more

Auto Trait Implementations

```
impl<C, H> Freeze for Prover<C, H>
```

```
impl<C, H> RefUnwindSafe for Prover<C, H>
```

```
where
    C: RefUnwindSafe,
    H: RefUnwindSafe,
```

```
impl<C, H> Send for Prover<C, H>
```

```
impl<C, H> Sync for Prover<C, H>
```

```
impl<C, H> Unpin for Prover<C, H>
```

```
where
    C: Unpin,
    H: Unpin,
```

```
impl<C, H> UnwindSafe for Prover<C, H>
```

```
where
    C: UnwindSafe,
    H: UnwindSafe,
```

Blanket Implementations

Source

```
impl<T> Any for T

where
    T: 'static + ?Sized,
```

Source

```
impl<T> Borrow<T> for T

where
    T: ?Sized,
```

Source

```
impl<T> BorrowMut<T> for T

where
    T: ?Sized,
```

Source

```
impl<T> CloneToUninit for T

where
    T: Clone,
```

Source

```
impl<T> Conv for T
```

Source

```
impl<T> FmtForward for T
```

Source

```
impl<T> From<T> for T
```

Source

```
impl<T> Instrument for T
```

Source

```
impl<T, U> Into<U> for T

where
    U: From<T>,
```

Source

```
impl<T> IntoEither for T
```

Source

```
impl<T> Pipe for T

where
    T: ?Sized,
```

Source

```
impl<T> Pointable for T
```

Source

```
impl<T> Same for T
```

Source

```
impl<T> Tap for T
```

Source

§

```
impl<T> ToOwned for T

where
    T: Clone,
```

Source

```
impl<T> TryConv for T
```

Source

```
impl<T, U> TryFrom<U> for T

where
    U: Into<T>,
```

Source

```
impl<T, U> TryInto<U> for T

where
    U: TryFrom<T>,
```

Source

```
impl<V, T> VZip<V> for T

where
    V: MultiLane<T>,
```

Source

```
impl<T> WithSubscriber for T
```

commonware_consensus::simplex

Enum Error Copy item path

Settings

Help

Summary

Source

```
pub enum Error {
    NetworkClosed,
    InvalidMessage,
    InvalidContainer,
    InvalidSignature,
}
```

Errors that can occur during consensus.

Variants

NetworkClosed

InvalidMessage

InvalidContainer

InvalidSignature

Trait Implementations

Source

impl Debug for Error

Source

fn fmt(&self, f: &mut Formatter<'_>) -> Result

Formats the value using the given formatter. Read more

Source

impl Display for Error

Source

```
fn fmt(&self, __formatter: &mut Formatter<'_>) -> Result
```

Formats the value using the given formatter. Read more

Source

impl Error for Error

1.30.0 · Source

```
fn source(&self) -> Option<&(dyn Error + 'static)>
```

Returns the lower-level source of this error, if any. Read more

1.0.0 · Source

```
fn description(&self) -> &str
```

👎Deprecated since 1.42.0: use the Display impl or to_string()

Read more

1.0.0 · Source

```
fn cause(&self) -> Option<&dyn Error>
```

👎Deprecated since 1.33.0: replaced by Error::source, which can support downcasting

Source

```
fn provide<'a>(&'a self, request: &mut Request<'a>)
```

🔬This is a nightly-only experimental API. (`error_generic_member_access`)

Provides type-based access to context intended for error reports. Read more

Auto Trait Implementations

impl Freeze for Error

impl RefUnwindSafe for Error

```
impl Send for Error
```

```
impl Sync for Error
```

```
impl Unpin for Error
```

```
impl UnwindSafe for Error
```

Blanket Implementations

Source

```
impl<T> Any for T

where
    T: 'static + ?Sized,
```

Source

```
impl<T> Borrow<T> for T

where
    T: ?Sized,
```

Source

```
impl<T> BorrowMut<T> for T

where
    T: ?Sized,
```

Source

```
impl<T> Conv for T
```

Source

```
impl<T> FmtForward for T
```

Source

```
impl<T> From<T> for T
```

Source

```
impl<T> Instrument for T
```

Source

```
impl<T, U> Into<U> for T

where
    U: From<T>,
```

Source

```
impl<T> IntoEither for T
```

Source

```
impl<T> Pipe for T

where
    T: ?Sized,
```

Source

```
impl<T> Pointable for T
```

Source

```
impl<T> Same for T
```

Source

```
impl<T> Tap for T
```

Source

§

```
impl<T> ToString for T

where
    T: Display + ?Sized,
```

Source

```
impl<T> TryConv for T
```

Source

```
impl<T, U> TryFrom<U> for T

where
    U: Into<T>,
```

Source

```
impl<T, U> TryInto<U> for T

where
    U: TryFrom<T>,
```

Source

```
impl<V, T> VZip<V> for T

where
    V: MultiLane<T>,
```

Source

```
impl<T> WithSubscriber for T
```